Marked-up version of amended claims to show changes:

1. (Amended) A method for managing connection requests to a pool of servers identified by a given URL, comprising the steps of:

in response to a connection request from a given client machine that initiates a session, associating a session identifier with a given server in the pool;

using the session identifier in a redirection response;

returning the redirection response to the given client to redirect the connection request to the given server; and

during the session, receiving at ~~redirecting to~~ the given server any additional connection requests from the given client machine.

2. The method as described in claim 1 wherein the step of using the session identifier includes generating a virtual URL.

3. The method as described in claim 2 wherein the virtual URL comprises a URL in the connection request modified to include the session identifier.

4. The method as described in claim 1 wherein the session identifier is incorporated in data returned from the given server to the client machine.

5. The method as described in claim 1 further including the step of:

in response to a connection request from the given client machine that terminates the session, inactivating the session identifier.

6. The method as described in claim 1 wherein the given client machine include a browser.

7. The method as described in claim 1 wherein each of the servers in the pool supports a similar set of objects.

8. The method as described in claim 1 wherein the session identifier is associated with a given server as a function of a load balancing protocol.

9. (Amended) A method for managing connection requests to a pool of servers, comprising the steps of:

responsive to a connection request~~s~~ from a client machine~~s~~ to ~~that~~ initiate a user session~~s~~, associating a ~~each~~ user session originating from a client machine with a given server in the pool in accordance with a load balancing protocol; ~~and~~

returning a redirection response to the client machine for the connection request; and

during the user session, receiving at ~~redirecting to~~ the given server any additional connection requests originating from the client machine.

10. The method as described in claim 9 wherein the associating step comprises:

generating a virtual URL by modifying a given URL to include a session identifier;

using the virtual URL to redirect the connection request to the given server.

11. The method as described in claim 10 further including the step of:

inactivating the virtual URL upon completion of the user session.

12. The method as described in claim 10 wherein all data returned from given server to the client machine includes the session identifier.

13. The method as described in claim 9 wherein each of the servers in the pool supports a similar set of given objects.

14. The method as described in claim 9 wherein each client machine include a Web browser.

15. (Amended) A computer program product in a computer-readable medium for managing connection requests to a pool of servers, comprising the steps of:

means~~,~~ responsive to a connection request~~s~~ from a client machine~~s~~ to ~~that~~ initiate a user session~~s~~~~,~~ for associating a ~~each~~ user session originating from a client machine with a given server in the pool in accordance with a load balancing protocol;~~ and~~

means for returning a redirection response to the client machine for the connection request; and

means operative during the ~~each~~ user session for receiving at ~~redirecting to~~ the given server any additional connection requests originating from the client machine.

16. The computer program product as described in claim 15 wherein the associating means comprises:

means for generating a virtual URL by modifying a given URL to include a session identifier;

means for redirecting a given connection request to the given server using the virtual URL.

17. The computer program product as described in claim 16 further including:

means for inactivating the virtual URL upon completion of the user session.

18. (Amended) A server for managing a pool of servers at a Web site identified by a given URL, comprising:

a processor;

an operating system;

a load balancing routine; and

a redirector routine for managing HTTP connection requests to the Web site, comprising:

means responsive to a connection request~~s~~ from a client machine~~s~~ to ~~that~~ initiate a user session~~s~~ for associating a ~~each~~ user session originating from a client machine with a given server in the pool in accordance with the load balancing routine; ~~and~~

means for returning a redirection response to the client machine for the connection request; and

means operative during the ~~each~~ user session for redirecting to the given server any additional connection requests originating from the client machine.

19. The server as described in claim 18 wherein the means for associating comprises:

means for generating a virtual URL by modifying a given URL to include a session identifier;

means for redirecting a given connection request to the given server using the virtual URL.

20. The server as described in claim 19 wherein the redirector further includes:

means for inactivating the virtual URL upon completion of the user session.

21. (Amended) A method of managing a pool of servers at a Web site identified by a given URL, comprising the steps of:

responsive to a connection request~~s~~ from a client machine~~s~~ to ~~that~~ initiate a user session~~s~~, associating a ~~each~~ user session originating from a client machine with a server in the pool of servers~~,~~ in order to distribute~~ing~~ ~~the~~ user sessions across the pool of servers in accordance~~ing~~ with ~~to~~ a load balancing protocol;~~ and~~

returning a redirection response to a given client machine for the connection request; and

during a given user session initiated from the ~~a~~ given client machine, serving content to the given client machine only from its associated server.

22. (Amended) A method of managing a pool of servers at a Web site identified by a given URL, comprising the steps of:

in response to a connection request containing the given URL from a given client machine that initiates a session, associating a session identifier with a given server in the pool of servers;

generating a virtual URL by modifying the given URL from the connection request to include the session identifier;

generating a redirection response comprising the virtual URL; and

sending the redirection response to the given client machine to redirect the connection request to the given server

~~during each user session initiated from a given client machine, temporarily redirecting all connection requests originating from the client machine to a given server in the pool; and~~

~~distributing the user sessions across the pool of servers according to a load balancing protocol~~.

REMARKS

Claims 1-22 are currently pending in the present application. Claims 1-22 were rejected in the previous Office action. In this response, independent claims 1, 9, 15, 18, 21, and 22 have been amended; no claims have been canceled. Reconsideration of the claims is respectfully requested.

I. Summary of Present Invention

A primary object of the present invention is to provide a method for distributing client requests across a pool of servers on a per-session basis rather than on a per-connection basis. Preferably, a given server in the pool of servers is allocated a given number of sessions such that a client's HTTP connection requests are handled by the same server throughout a user session. Another object of the present invention is to implement a load balancing routine across a set of servers while ensuring that connection requests from a particular client during its session are still serviced by the same server in the pool of servers.

In response to a connection request from a client that initiates a user session, a front-end managing server intercepts the request and recognizes that the connection request will initiate a user session. The managing server can be viewed as acting as a redirector for connection requests; the managing server may query a load balancing routine to determine which server in the pool of servers should service the new session. A unique session identifier is associated with a given server in the pool of servers, and the session identifier is then incorporated into a base URL for the assigned server, thereby forming a "virtual URL" that is returned in an appropriate redirection response to the client. The client then automatically issues a new HTTP connection

request using the newly generated virtual URL. All subsequent data that is returned to the client will incorporate the virtual URL such that subsequent requests from the client will contain the session identifier as part of the URL. Requests from the client to the assigned server are then routed to the appropriate server in accordance with the URL, and the appropriate server can use the session identifier to associate the request with a user session.

At some point in time, the user may perform some type of action that indicates that the session is being terminated, such as requesting a Web page for a logoff operation. The session identifier for the user session is then inactivated in an appropriate manner, and the managing server releases the assigned server from its association with the client.

II. 35 U.S.C. § 103(a)—Obviousness—*Bayeh et al.* in view of *Wallis* and further in view of *Brood et al.*

The Office action has rejected claims 1-4, 6-10, 12-16, 18, 19, 21, and 22 under 35 U.S.C. § 103(a) as unpatentable over Bayeh et al., "Maintaining Sessions in a Clustered Server Environment", U.S. Patent Number 6,098,093, issued 08/01/2000, in view of Wallis, "Load Balancing of Connections to Parallel Servers", U.S. Patent 5,740,371, issued 08/14/1998. This rejection is respectfully traversed.

In its background section, Bayeh et al. explains that session identifiers have been implemented within HTTP communications as a method for managing state information, even though HTTP is defined as a stateless protocol. There have been two primary approaches: cookies and URL rewriting. In the first case, a server can generate a cookie in response to a client request, and the cookie contains a session identifier. The session cookie is passed back to the client,

and the cookie is returned by the client with each subsequent request to the server. When the server receives a request with a cookie, the session identifier within the cookie enables the server to find information about previous transactions for the client, thereby allowing the server to maintain state information about the client. In the second approach, URL rewriting is used to ensure that requests sent to the server will have the session identifier in the URL of a request. When a Web page is generated in response to a client request, the hypertext links that are embedded in the Web page are modified to contain the session identifier for the requesting client. When a user at the client clicks on one of these hypertext links, the URL in the request that is returned to the server will contain the client's session identifier.

Bayeh et al. teaches a system in which session-related state information is managed in a clustered server environment. A load-balancing host acts as a front-end processor for the cluster of servers to route client requests to the servers in accordance with a load-balancing routine at the host. The system taught by Bayeh et al. solves the problem of extending session support across a cluster of servers by providing session services via a set of plug-in servlet engines. One of these servlet engines will be installed on each Web server in the cluster of servers, and one of the servlet engines is configured to function as a session management server while the other servlet engines are configured to function as session clients. When an HTTP request is received from a client, the request is sent by the load-balancing host to one of the Web servers. The Web server then passes the request to the plug-in servlet engine based on certain criteria, such as the presence of a servlet-identifying string as part of the host destination

address in the URL, thereby allowing the server to route the client request to a specific servlet engine.

Bayeh et al. then continues to describe the system as follows in column 10, lines 10-31:

> When the plug-in servlet engine 72 gets the request 112, the request may or may not include a session identifier for a session with this client. If this is the first request of a new session, no session ID will be present. If this is a request of an existing session, then there will be a session ID included using either the cookie mechanism or URL rewriting, as discussed earlier. In the preferred embodiment, the servlet engine passes this request on to the proper servlet (that is, the one identified by the syntax of the URL in the HTTP request) by invoking the servlet's "service" method, which is the standard API used to communicate with a servlet according to the Java Servlet "HttpServletRequest" API definition. (Changes may be required before passing this request to the servlet. For example, if URL rewriting is used, the servlet engine will strip the session ID form the address in the URL before the request is forwarded.) The service method has a request object and a response object as parameters. These objects encapsulate data sent to and from the servlet 92. The servlet engine may make changes to the request before forwarding it to the servlet, for example by modifying the URL so that the proper servlet is identified by the address in the URL.

Bayeh et al. also describes the manner in which the system concludes a session at column 11, lines 3-8:

> When the servlet processing is finished, the session object is returned to the session pool, where it can be accessed for subsequent transactions with this servlet or a different servlet in the clustered environment. In this way, the state of the session can be communicated among the clustered servers and their servlets.

As can be understood from the description above, a client request is received by a load-balancing host, which then forwards the client request to a Web server in a cluster of Web servers in accordance with a load-balancing algorithm. Assuming that the client request does not contain a session

identifier using either a cookie or URL rewriting technique, then the plug-in servlet client at the receiving Web server coordinates with the plug-in servlet server to generate a session identifier for the user session associated with the client request. Subsequent requests from the client during the same user session will contain the session identifier, and the session identifier can be used to manage information for the user session across many client requests or connections.

Three important distinctions can be made between the system of the present invention and the system described by Bayeh et al.. In the present invention, the front-end managing server participates in the session management, and client requests are redirected from the front-end managing server back through the client to a server in the pool of servers. In addition, the present invention ensures that the same server in the pool of servers receives all of the client requests for a particular user session. In contrast, the load-balancing host in the system taught by Bayeh et al. does not participate in the session management services among the cluster of Web servers, and the client request is forwarded directly from the load-balancing host to one of the Web servers. In addition, once a session identifier has been assigned to a user session in the system taught by Bayeh et al., a client request for the user session may be received at any of the servers in the cluster of servers because the plug-in servlet clients at each of the servers coordinate the session management information amongst themselves with the assistance of the plug-in servlet server. In other words, there is no guarantee that the same server will process all client requests for a given user session. Hence, the load on the cluster of servers is balanced on a per-connection basis, as is recognized by the rejection.

With respect to each of the independent claims 1, 9, 15, 18, 21, and 22, the rejection admits that "Bayeh does not teach redirecting to the given server an additional request as claimed."--(Office action, page 2, last paragraph). In order to remedy this deficiency, the rejection relies on the following two portions of Wallis. The first portion from the background of Wallis describes dynamic per-session load balancing at column 2, lines 18-43:

> There are several approaches that have been developed which attempt to balance the load on processors of a parallel server. A brief taxonomy of these includes:
>
> (a) Static balancing
> (b) Dynamic balancing
> (i) per-session (session granularity)
> (ii) migratable (sub-session granularity)
>
> Static balancing requires the administrator of the server to estimate the distribution of loads across the parallel server and attempt to even out the load via permanent associations with clients (such as terminals) and the processors of the parallel server. This is appropriate in some circumstances but has obvious disadvantages when compared to dynamic systems which will adapt to the changing loads on the server processors.
>
> Dynamic load balancing can be achieved by using a 'per-session' load balancing technique. This type of dynamic load balancing is performed at creation of a session or submission of a job, and the affiliation between the client machine and the server lasts for the duration of the session or job. Currently research is being carried out into 'migratable' dynamic load balancing techniques, where jobs can be migrated from one server processor to another at any stage during their processing, in response to a change in load across the processors of the server.

The second portion from the abstract of Wallis teaches a specific system for dynamic per-session load balancing:

> A system and method for facilitating compatibility with a prior process used for connecting a user terminal to a selected server in a system having a plurality of servers. The prior process has the server perform a task on behalf of the terminal. The system includes a server determination process for retrieving from storage a table of data identifying the servers, and for determining an

address currently associated with each server. Then, a chooser process enables the user to select from the table, one of said servers for connection to the terminal with a connection process, responsive to a signal from the terminal, that initiates a connection of the terminal to the selected server. The plurality of servers in the system includes at least one parallel server comprised of multiple processors. The data in the table identifying the parallel server is a generic identifier, and a process is employed, which preferably uses predetermined dynamic criteria, to periodically associate a specific processor of the parallel server with that generic name. To enable the load to be balanced across the plural processors of the parallel server, the server determination process is adapted to recognize any generic identifier in the table, and to determine the address associated with that generic identifier after a user selection of the parallel server corresponding to that generic identifier has been made.

As should be apparent from these passages, Wallis discloses associations between clients and servers on a session basis. However, the communication method of Wallis is not similar at all to the present invention nor the system disclosed in Bayeh et al.. In the system disclosed in Wallis, the client, i.e. user terminal, communicates with an "adapter chooser" to determine which server should be contacted, and the client contacts this server to establish a connection which lasts for the client's entire session. After the initial connection is made, the client may then send processing requests to the server. This type of communication process differs substantially from the stateless communication protocols of the present invention and of Bayeh et al. in which a client may simply send a request to a Web server, and the initial request can be regarded as initiating a user session; in other words, a special "initialization" state is not required. Hence, it would not be possible to modify Wallis to include the redirection response of the present

invention without rendering Wallis inoperable for its intended use.

Although Wallis discloses that client-server associations can be distributed across a set of servers on a per-session basis, it appears that the rejection relies on Wallis only for this feature because Wallis clearly does not disclose the feature of redirection of client requests as required by the present invention nor the routing of client requests in a manner similar to that disclosed by Bayeh et al.. Hence, the rejection can be summarized as arguing that Bayeh et al. discloses the routing of client requests on a per-connection basis and that Wallis discloses server support for connecting a client to a server on a per-session basis.

However, this combination of features does not address the fact that the operations that are discussed in Bayeh et al. for routing client requests are not equivalent to the redirection operations of the present invention, thereby resulting in at least three important differences as discussed above. While the rejection argues that Bayeh et al. can be combined with Wallis, the rejection merely asserts that the combination produces redirection of client requests on a per-session basis without explaining which features in the system disclosed by Wallis would specifically be incorporated into the system disclosed by Bayeh et al.. In other words, the rejection merely asserts that a specific effect or advantage described in Wallis, i.e. per-session management, could be achieved in the method or system described by Bayeh et al. without specifically describing the manner by which this combined system or method would operate.

In order to clarify the claim language in the present application with respect to the applied references, most specifically Bayeh et al., Applicant has amended each of the

independent claims to clearly recite the feature of "returning a redirection response to the client machine for the connection request" using this phrase or similar language. While Bayeh et al. discloses the routing of client requests through a cluster of servers, Bayeh et al. does not disclose that the response from the server to the client request is a "redirection response", which is a specific term of art with respect to network-related protocols such as HTTP. As mentioned above, the front-end managing server in the present invention redirects the client request back through the client through the use of a redirection response.

Neither Bayeh et al. nor Wallis disclose the use of a redirection response. Moreover, the systems disclosed in either Bayeh et al. or Wallis would be clearly inoperable if this feature were incorporated into either of these systems or into a hypothetical system in which features from both systems were combined together, as argued by the rejection.

Applicant has also amended the language of some of the independent claims so that it is clear that the present invention is applicable in a system with a single client machine that is being used for a single user session and does not require multiple client machines nor multiple user sessions. For example, with respect to independent claim 9, "client machines" has been changed to "a client machine", and "user sessions" has been changed to "a user session". Applicant asserts that these amendments have not been made for the purposes of patentability in any manner whatsoever, particularly not for obviating prior art.

With the amendment to the claims provided by this response to the Office action, Bayeh et al. clearly fails to show a feature of the present invention as currently claimed and as asserted by the Office action, thereby rendering Bayeh

et al. incapable of being used as a primary reference as argued by the current rejection. Moreover, Wallis and the combination of Wallis and Bayeh et al. fail to show this feature. As should be recognized, because both the primary and secondary references in the rejection fail to disclose the claimed features against which the references were applied, and because the references fail to be combinable to produce this feature, the rejection fails to fulfill the requirements of a proper obviousness argument. The examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Only when a *prima facie* case of obviousness is established does the burden shift to the applicant to produce evidence of nonobviousness. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Rijckaert*, 9 F.3d 1531, 1532, 28 U.S.P.Q.2d 1955, 1956 (Fed. Cir. 1993). If the Patent Office does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985). In response to an assertion of obviousness by the Patent Office, the applicant may attack the Patent Office's *prima facie* determination as improperly made out, present objective evidence tending to support a conclusion of nonobviousness, or both. *In re Fritch*, 972 F.2d 1260, 1265, 23 U.S.P.Q.2d 1780, 1783 (Fed. Cir. 1992).

With respect to independent claims 1, 9, 15, 18, 21, and 22, Applicant respectfully submits that the applied references cannot be combined nor modified to produce the claimed

invention. Hence, a rejection of the independent claims cannot be based upon the cited prior art to establish a *prima facie* case of obviousness. Therefore, a rejection of the independent claims under 35 U.S.C. § 103(a) has been shown to be insupportable in view of the cited prior art, and the independent claims are patentable over the applied references. Applicant respectfully requests the withdrawal of the rejection of the independent claims. Applicant further argues that all of the pending claims, including the dependent claims which comprise the elements of their independent claims by inclusion, are distinguishable over Bayeh et al. in view of Wallis for these reasons, and Applicant kindly requests the withdrawal of the rejection of all claims.

Applicant further notes that claims 5, 11, 17, and 20 were rejected under 35 U.S.C. § 103(a) as unpatentable over Bayeh et al., Wallis, and further in view of Brodd et al., "Network Communications Interface", U.S. Patent 5,303,238, issued 04/12/1994. The rejection states that the combination of Bayeh et al. and Willis does not teach the inactivation of a session identifier, and the rejection of these claims then relies on Brodd et al. as disclosing the inactivation of a session identifier. However, as can be seen in one of the passages that have been recited above, Bayeh et al. does state that "the session object is returned to the session pool, where it can be accessed for subsequent transactions ...".

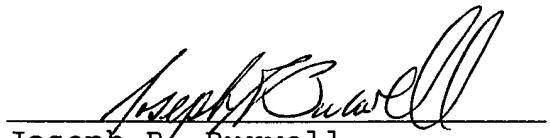
III. Conclusion

It is respectfully urged that the present patent application is patentable, and Applicant kindly requests a Notice of Allowance.

For any other outstanding matters or issues, the examiner is urged to call or fax the below-listed telephone numbers to expedite the prosecution and examination of this application.

DATE: January 29, 2002

Respectfully submitted,



Joseph R. Burwell
Reg. No. 44,468
ATTORNEY FOR APPLICANT

Law Office of Joseph R. Burwell
P.O. Box 28022
Austin, Texas 78755
(512) 502-9448 (voice)
(512) 597-1218 (fax)